VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

NUMERICAL SOLUTIONS OF EMC PROBLEMS OF SMALL AIRPLANES

NUMERICKÁ ŘEŠENÍ PROBLEMATIKY EMC MALÝCH LETADEL

ZKRÁCENÁ VERZE DIZERTAČNÍ PRÁCE
SHORT VERSION OF DOCTORAL THESIS

AUTOR PRÁCE Ing. VLADIMÍR ŠEDĚNKA
AUTHOR

VEDOUCÍ PRÁCE prof. Dr. Ing. ZBYNĚK RAIDA
SUPERVISOR

BRNO 2013

ABSTRACT

The dissertation describes actual problems in certifying small airplanes, which are to be solved by numerical modeling of the airplanes. The work is closely linked to the European project HIRF-SE, which deals with this problem. The essential part of the work is devoted to describing design of two modules within the HIRF-SE framework: time-domain finite-element solver BUTFE and its excitation module BUTFE_EXC. The thesis describes solution of absorbing boundary conditions, dispersive media, anisotropy and thin wire approximation. Special attention is devoted to a proper approximation of thin wires with sharp bends. Current implementation of the approximation leads to overlaps of wire segments.

KEYWORDS

electromagnetic compatibility, finite element method, high intensity radiated pulse, HIRF-SE, Amelet-HDF, HDF, absorbing boundary condition, dispersion, anisotropy, thin wire approximation

ABSTRAKT

Disertace popisuje současné problémy v certifikaci malých letadel, které by se měly v budoucnu řešit numerickým modelováním. Tento postup má zefektivnit návrh a zlevnit certifikaci letadel. Práce je úzce spjata s projektem HIRF-SE, který se problematikou certifikace letadel numerickými metodami zabývá. Podstatná část práce je věnována popisu dvou modulů pro platformu HIRF-SE: řešič BUTFE založený na metodě konečných prvků v časové oblasti a budicí nástroj BUTFE_EXC. Práce popisuje řešení pohlcujících okrajových podmínek, modelování disperzních a anizotropních materiálů a aproximaci tenkých drátů. Speciální pozornost je věnována řešení aproximace tenkých drátů s ostrými ohyby, jejíž současná formulace způsobuje překryvy mezi jednotlivými segmenty drátu.

KLÍČOVÁ SLOVA

elektromagnetická kompatibilita, metoda konečných prvků, vyzařované pole s vysokou intenzitou, HIRF-SE, Amelet-HDF, HDF, pohlcujících okrajové podmínky, disperze, anizotropie, aproximace tenkého drátu

DECLARATION

I declare that I have written my doctoral thesis on the theme of "Numerical solutions of EMC problems of small airplanes" independently, under the guidance of the doctoral thesis supervisor and using the technical literature and other sources of information which are all quoted in the thesis and detailed in the list of literature at the end of the thesis.

As the author of the doctoral thesis I furthermore declare that, as regards the creation of this doctoral thesis, I have not infringed any copyright. In particular, I have not unlawfully encroached on anyone's personal and/or ownership rights and I am fully aware of the consequences in the case of breaking Regulation § 11 and the following of the Copyright Act No. 121/2000 Coll., and of the rights related to intellectual property right and changes in some Acts (Intellectual Property Act) and formulated in later regulations, inclusive of the possible consequences resulting from the provisions of Criminal Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

Brno ....

(author's signature)

ACKNOWLEDGEMENTS

The work described in this thesis has received funding from EC FP7 under grant no. 205294 (the HIRF SE project).
Further financing was provided by the Czech Ministry of Education (the grant no. 7R09008).

The described research was performed in laboratories supported by the SIX project; the registration number CZ.1.05/2.1.00/03.0072, the operational program Research and Development for Innovation.

A support of the project CZ.1.07/2.3.00/20.0007 Wireless Communication Teams financed by the operational program Education for Competitiveness is also gratefully acknowledged.

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

This research used resources of the National Supercomputing Center IT4Innovations, which is supported by the Op VaVpI project number CZ.1.05/1.1.00/02.0070.

CONTENTS

1 Introduction 6

2 TDFE solver 7

2.1 Connecting the modules within the framework 7

2.2 Description of the modules 10

2.3 Mathematical formulation 12

2.4 FEM approximation 14

2.5 Conclusion 15

3 Bent wires 16

3.1 Proposed approach 16

3.2 Construction strategies 17

3.3 Conclusion 18

4 File interface 19

4.1 Module BUTFE_EXC 19

4.2 Module BUTFE 19

4.3 Amelet-HDF C library 20

4.4 Conclusion 21

5 Development tools 22

5.1 CMP 22

5.2 Visualizer 22

5.3 Conclusion 24

6 Conclusion 25

Bibliography 27

1 INTRODUCTION

A growing involvement of electronic devices inside an aircraft and especially employment of electrical devices in critical flight systems makes *electromagnetic compatibility* (EMC) a more and more important part of the aircraft design process. Internal electronic systems interfere with each other. In addition, all the systems are exposed to influences of radio and TV transmitters, satellite transmitters, radars, lightning strikes, etc. Such an environment is known as *High-Intensity Radiated Fields* (HIRF).

Beyond the usage of electronic devices inside the aircraft, growing interest in EMC is also given by an effort to reduce costs and increase in-flight comfort. Conductive parts are increasingly being replaced by composite materials which makes the aircraft lighter but also more vulnerable to HIRF.

Before final production, each aircraft must pass various EMC tests. This typically consists in simulating the airplane (or its model) at an EMC test site trying to model the worst case scenario. These tests are expensive in terms of space, time and money. Sometimes they may be also destructive.

Previously mentioned inconveniences lead to the effort of numerical modeling before testing. Testing costs are replaced by computational costs which are decreasing due to advances in computer technology. This improves the efficiency of the design process and makes the certification cheaper.

Although various software tools for the *electromagnetic* (EM) field analysis are already developed, they usually do not cover the whole area of possible EMC problems related to the aircraft. Moreover, some differences can appear in results obtained from software based on different numerical methods due to the strengths and weaknesses of the particular method. Basically, user should pick the right method based on his knowledge.

Project *High-Intensity Radiated Fields - Synthetic Environment* (HIRF-SE) [1] aimed to create a numerical modeling computer framework for the aeronautic industry. The framework should improve the design phase of the aircraft and also reduce costs of the EMC tests.

Each partner was developing a specific module making use of the particular method. All the modules were connected with each other via the format Amelet-HDF [2], which was also being developed within the project.

The dissertation is closely related to HIRF-SE and focuses on the following topics:

- **TDFE solver:** creating a three-dimensional *time-domain finite element* (TDFE) solver aiming at EM simulations
- **Bent wires:** improve the solver's capabilities towards solving EMC problems concerning small airplanes (thin wire approximation of bent wires)
- **File interface:** design an input/output wrapper that will adapt the TDFE solver for application in the HIRF-SE framework
- **Development tools:** create development tool(s) for simplifying the design and test phase and facilitating further improvements of the TDFE solver

2 TDFE SOLVER

This chapter is dedicated to explaining the core functionality of the TDFE solver called BUTFE and its excitation module BUTFE_EXC. The involvement of the solver and the excitation module within the HIRF-SE framework is shown in chapter 2.1. The semantic location within the framework is shown on a sample simulation. A description of both modules from the user's point of view follows in chapter 2.2. Chapter 2.3 focuses on the mathematical formulation of the problem. Types of basis functions involved in the solver are described in chapter 2.4. Chapter 2.5 concludes this topic.

2.1 Connecting the modules within the framework

This chapter describes work in the HIRF-SE framework from the user's perspective. The entire simulation is formed by a chain of sub-simulations performed by various modules. For each sub-simulation, the HIRF-SE framework prepares an input Amelet-HDF file, executes the main executable of a given module and reads the outputs from an output Amelet-HDF file created by the module. A schematic diagram of each sub-simulation shows inputs (left side), module used for the simulation (middle) and outputs (right side). Notation "(a..b)" represents minimal, resp. maximal allowed amount of data of a given data type.

Mesh generation

The first sub-simulation is responsible for mesh generation. It uses the module GiD, which was developed by the group CIMNE [13]. Figure 2.1 shows that the module accepts up to five different types of input data and can produce up to two types of output data. All of them are optional.

In this case, the module is used as a mesh generator. A window, opened within the sub-simulation, allows the user to define the geometry, generate an output mesh and store it into an object *mesh_gid*.

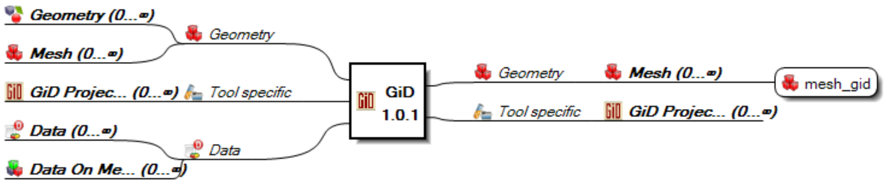


Fig. 2.1 *The HIRF-SE framework: Mesh generation.*

Sources determination

The previously generated mesh *mesh_gid* can be linked with the excitation module BUTFE_EXC (see figure 2.2). The module requires one object of type *Mesh* and produces 3 types of output data: *Mesh*, *Global environment* and one or more objects of type *Electromagnetic source*.

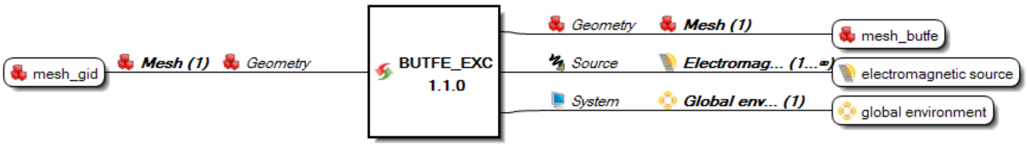


Fig. 2.2 *The HIRF-SE framework: Sources determination.*

Based on the user defined excitation(s), the module slightly modifies the input *mesh_gid* into *mesh_butfe* and creates a list of requested time samples *global environment* and excitation samples *electromagnetic source*. Detailed specification can be found in [2].

Computation

This sub-simulation employs the TDFE solver module BUTFE (see figure 2.3). The module expects *Mesh*, *Global environment* and *Electromagnetic source(s)* from the BUTFE_EXC module. The user has to define the remaining input data of the simulation: materials and physical quantities to compute. Materials are set via object *Link* and objects of the *Material* category. The module accepts predefined materials like vacuum, *perfectly electric conductor* (PEC), *perfect magnetic conductor* (PMC), classical materials and dispersive materials given by the Debye model. The materials can be either isotropic or anisotropic defined by a material tensor.

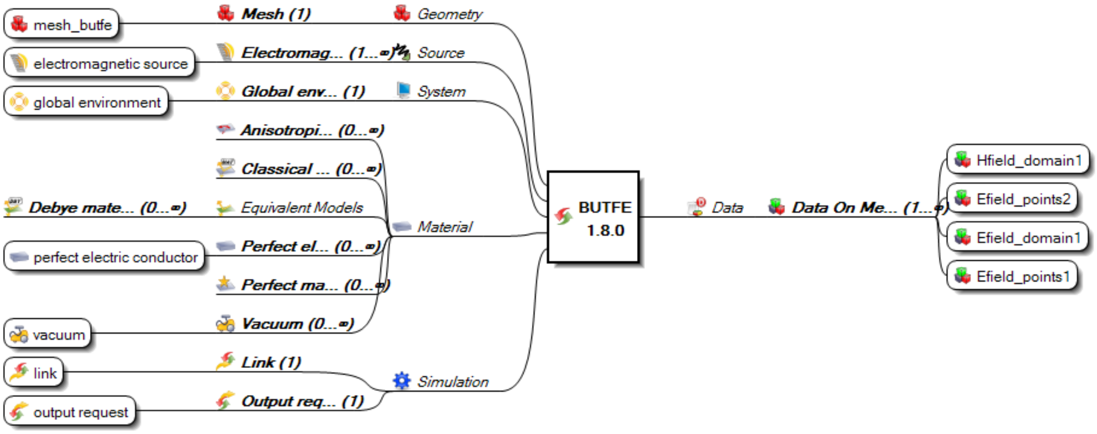


Fig. 2.3 *The HIRF-SE framework: EM field computation.*

The requested physical quantities are defined via *Output request*. The BUTFE module supports the following output quantities:

- electric and/or magnetic field intensity in the middle of a given tetrahedron,
- electric and/or magnetic field intensity in a predefined point,
- electric current flowing through a predefined wire in the middle of wire segments.

Only an electric/magnetic field in the domain and electric field in two specific groups of points are requested in this example.

Note that the BUTFE module solves the intensity of the electrical field. The intensity of the magnetic field is obtained using post-processing.

Visualization

The last part of the simulation chain could be visualization. The module Paraview is developed by the company AxesSim. It is a wrapper for a known visualization program of the same name developed by Kitware, Inc. [14].

Another visualization tool (to be considered as the main tool) in the HIRF-SE framework is GiD.

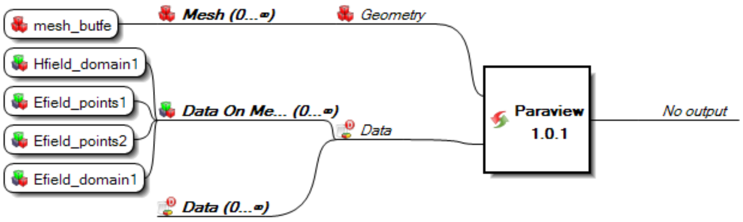


Fig. 2.4 *The HIRF-SE framework: EM field visualization by Paraview.*

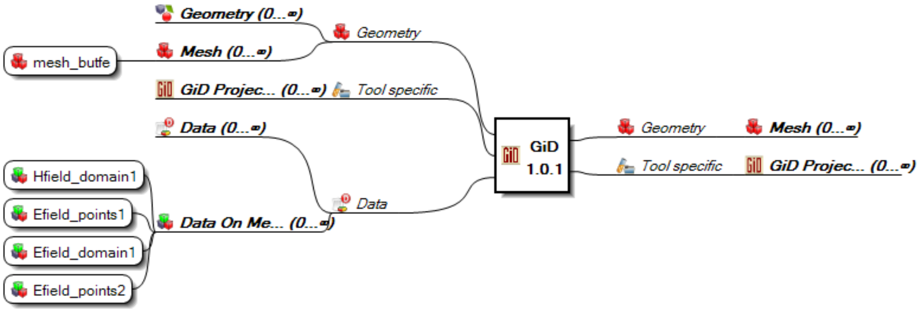


Fig. 2.5 *The HIRF-SE framework: EM field visualization by GiD.*

Figures 2.4 and 2.5 are only informative. The module Paraview can handle only inputs that cover all the elements of the geometry. In this case the Paraview would return that there are more points in the geometry than just the five predefined groups. The module GiD was not adapted to be compatible with output data of the BUTFE solver.

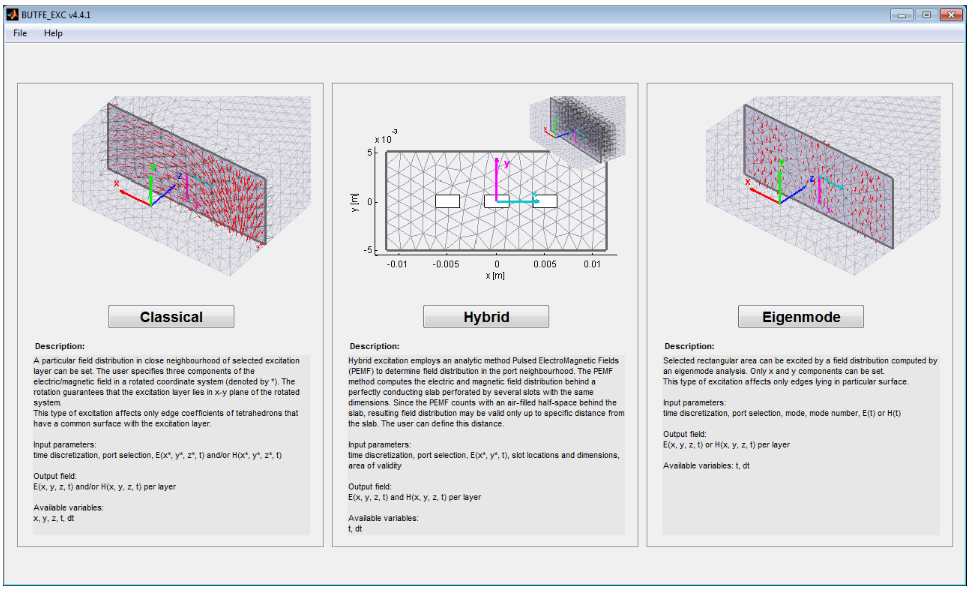
2.2 Description of the modules

BUTFE_EXC module

Excitation is closely connected with the mesh so in order to achieve easy operation, the module has to contain a *graphical user interface* (GUI). Based on my programming skills, I have decided to make this tool in the MATLAB© environment.

An executable for the HIRF-SE module was generated using MATLAB© Compiler™ [11]. A special set of libraries called *MATLAB Compiler Runtime* (MCR) package must be installed on a target machine in order to run the executable [12]. The MCR installation package can be freely distributed together with the module.

The excitation tool can work in three different modes: classical, hybrid and eigenmode. Classical excitation sets distribution in the nearest neighborhood of a given plane. Hybrid excitation computes the EM field behind a perforated slab using the *pulsed electromagnetic field* (PEMF) method [4] and uses the results for excitation. Eigenmode solves a 2D eigenmode analysis (using 2D FEM) of a given plane to get desired field distribution.



A particular field distribution in close neighbourhood of selected excitation layer can be set. The user specifies three components of the electric/magnetic field in a rotated coordinate system (denoted by *). The rotation guarantees that the excitation layer lies in x-y plane of the rotated system.
This type of excitation affects only edge coefficients of tetrahedrons that have a common surface with the excitation layer.

Input parameters:
time discretization, port selection, E(x*, y*, z*, t) and/or H(x*, y*, z*, t)

Output field:
E(x, y, z, t) and/or H(x, y, z, t) per layer

Available variables:
x, y, z, t, dt

Hybrid excitation employs an analytic method Pulsed ElectroMagnetic Fields (PEMF) to determine field distribution in the port neighbourhood. The PEMF method computes the electric and magnetic field distribution behind a perfectly conducting slab perforated by several slots with the same dimensions. Since the PEMF counts with an air-filled half-space behind the slab, resulting field distribution may be valid only up to specific distance from the slab. The user can define this distance.

Input parameters:
time discretization, port selection, E(x*, y*, t), slot locations and dimensions, area of validity

Output field:
E(x, y, z, t) and H(x, y, z, t) per layer

Available variables:
t, dt

Selected rectangular area can be excited by a field distribution computed by an eigenmode analysis. Only x and y components can be set.
This type of excitation affects only edges lying in particular surface.

Input parameters:
time discretization, port selection, mode, mode number, E(t) or H(t)

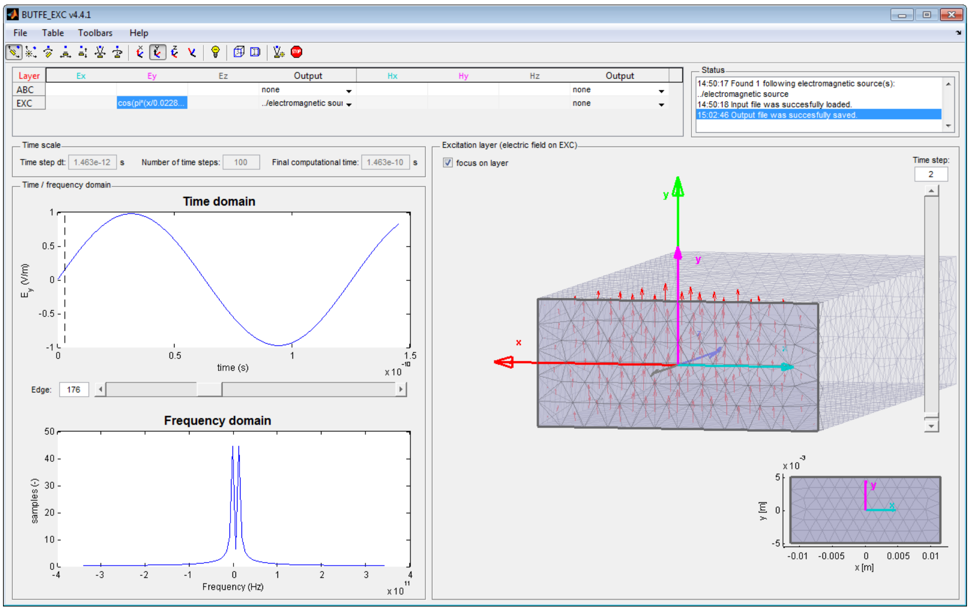
Output field:
E(x, y, z, t) or H(x, y, z, t) per layer

Available variables: t, dt

Fig. 2.6 *The excitation tool: choosing operation mode.*

All these three cases are restricted to using a plane layer from the input mesh. The layer is supposed to contain triangle elements. The tool extracts edges from tetrahedrons forming the neighborhood of the layer and adds the edges into the output mesh (that is the only difference between *mesh_gid* and *mesh_butfe*). The resulting excitation coefficients are related to the edges.

The general layout of the tool remains the same in the case of all the three modes. The tool contains the following sections: *Main table*, *Status*, *Time scale*, *Time domain* and *Excitation layer*. The *Main table* allows to set the excitation formula (time-domain dependency), *Status* informs about performed actions. Once the program starts, the input mesh is analyzed and the optimal time step is determined. The user can modify time step, number of steps or total computational time in the *Time scale* panel. The *Time* *domain* box displays selected excitation signal in both the time and frequency domain. The *Excitation layer* lets the user to check space distribution of the excitation. Geometry can be manipulated through tools in the *View* or *Camera* tool bar. Output data is written using the menu option *File→Save* (or by pressing CTRL+S). Figure 2.7 shows the layout of the BUTFE_EXC module.



Layer	Ex	Ey	Ez	Output	Hx	Hy	Hz	Output
ABC				none				none
EXC		cos(pi*(x/0.0228...		../electromagnetic sou				none

```
14:50:17 Found 1 following electromagnetic source(s):
../electromagnetic source
14:50:18 Input file was succesfully loaded.
15:02:46 Output file was succesfully saved.
```

Time step dt: 1.463e-12 s

Number of time steps: 100

Final computational time: 1.463e-10 s

☑ focus on layer

Time step: 2

Edge: 176

Fig. 2.7 *The excitation tool: classical.*

BUTFE module

The BUTFE module (TDFE solver) has no graphical interface. It only produces a text log, which is visible in the framework after the simulation.

Firstly, the input file is processed. The program can continue only if all required inputs are read successfully. The status of the read processes is reported in the log file. Information about the read process, sources, and applied simulation parameters is followed by brief information about the progress and total computational time.

Actual progress of the simulation can be monitored in the framework through the window "Follow simulation".

The module offers the possibility of a controlled end of the simulation from the framework. In this case, the output file will only contain results computed so far. Information about a premature end of the simulation is noted in the log file.

2.3 Mathematical formulation

The BUTFE solver is based on time domain Maxwell equations in differential form. The resulting system of equations (see equation 2.1) includes *absorbing boundary condition* (ABC), Debye media and thin-wire approximation of N wires.

$$
\begin{aligned}
\left[\frac{1}{(\Delta t)^2}\mathbf{A}+\frac{1}{2\Delta t}\mathbf{B}+\frac{1}{4}\mathbf{C}\right]\mathbf{v}^{\{n+1\}} &= \left[\frac{2}{(\Delta t)^2}\mathbf{A}-\frac{1}{2}\mathbf{C}\right]\mathbf{v}^{\{n\}} \\
&+\left[-\frac{1}{(\Delta t)^2}\mathbf{A}+\frac{1}{2\Delta t}\mathbf{B}-\frac{1}{4}\mathbf{C}\right]\mathbf{v}^{\{n-1\}} \\
+\left[\frac{1}{2\Delta t}\mathbf{D}+\frac{1}{4}\mathbf{E}\right]\mathbf{v}_{inc}^{\{n+1\}} &+\left[\frac{1}{2}\mathbf{E}\right]\mathbf{v}_{inc}^{\{n\}}+\left[-\frac{1}{2\Delta t}\mathbf{D}+\frac{1}{4}\mathbf{E}\right]\mathbf{v}_{inc}^{\{n-1\}} \\
&-\frac{1}{4}\mathbf{f}^{\{n+1\}}-\frac{1}{2}\mathbf{f}^{\{n\}}-\frac{1}{4}\mathbf{f}^{\{n-1\}} \\
&+\frac{1}{4}\mathbf{g}^{\{n+1\}}+\frac{1}{2}\mathbf{g}^{\{n\}}+\frac{1}{4}\mathbf{g}^{\{n-1\}} \\
&-\frac{1}{(\Delta t)^2}\mathbf{\Psi}_e^{\{n\}}
\end{aligned}
\tag{2.1a}
$$

where

$$
\mathbf{A}=\begin{bmatrix}\mathbf{T}^{\epsilon}+\mathbf{\Phi}^0 & 0 & \cdots & 0\\ 0 & \mathbf{T}_1^w & 0 & 0\\ \vdots & 0 & \ddots & 0\\ 0 & 0 & 0 & \mathbf{T}_N^w\end{bmatrix}
\qquad
\mathbf{D}=\begin{bmatrix}\mathbf{Q} & 0 & \cdots & 0\\ 0 & 0 & 0 & 0\\ \vdots & 0 & \ddots & 0\\ 0 & 0 & 0 & 0\end{bmatrix}
$$

$$
\mathbf{B}=\begin{bmatrix}\mathbf{T}^{\sigma}+\mathbf{Q} & \mathbf{P}_1^{w\top} & \cdots & \mathbf{P}_N^{w\top}\\ -\mathbf{P}_1^w & \mathbf{R}_1^w & 0 & 0\\ \vdots & 0 & \ddots & 0\\ -\mathbf{P}_N^w & 0 & 0 & \mathbf{R}_N^w\end{bmatrix}
\qquad
\mathbf{E}=\begin{bmatrix}\mathbf{P} & 0 & \cdots & 0\\ 0 & 0 & 0 & 0\\ \vdots & 0 & \ddots & 0\\ 0 & 0 & 0 & 0\end{bmatrix}
$$

$$
\mathbf{C}=\begin{bmatrix}\mathbf{S} & 0 & \cdots & 0\\ 0 & \mathbf{S}_1^w & 0 & 0\\ \vdots & 0 & \ddots & 0\\ 0 & 0 & 0 & \mathbf{S}_N^w\end{bmatrix}
\tag{2.1b}
$$

$$
\mathbf{\Psi}_e^{\{n\}}=\mathbf{\Phi}^{1/2}\begin{bmatrix}\mathbf{e}^{\{n\}}-2\mathbf{e}^{\{n-1\}}+\mathbf{e}^{\{n-2\}}\\ 0\\ \vdots\\ 0\end{bmatrix}+\sum_{p=1}^{P}e^{-\frac{\Delta t}{\tau_p}}\mathbf{\Psi}_e^{\{n-1\}}
$$

$$
\mathbf{f}=\begin{bmatrix}\tilde{\mathbf{f}}\\ 0\\ \vdots\\ 0\end{bmatrix}
\qquad
\mathbf{g}=\begin{bmatrix}0\\ \tilde{\mathbf{f}}_1^w\\ \vdots\\ \tilde{\mathbf{f}}_N^w\end{bmatrix}
\qquad
\mathbf{v}=\begin{bmatrix}\mathbf{e}\\ \mathbf{i}_1\\ \vdots\\ \mathbf{i}_N\end{bmatrix}
\qquad
\mathbf{v}_{inc}=\begin{bmatrix}\mathbf{e}_{inc}\\ 0\\ \vdots\\ 0\end{bmatrix}
$$

and

$$
\begin{aligned}
S_{i,j} &= \iiint\limits_V \Big[\nabla \times \mathbf{N}_i(\mathbf{r})\Big] \cdot \Big[\mathbf{R}^{\top^{-1}} \frac{1}{\mu_0 \hat{\mu}_r} \mathbf{R}^{-1} \nabla \times \mathbf{N}_j(\mathbf{r})\Big] \,\mathrm{d}V \\
T^{\epsilon}_{i,j} &= \iiint\limits_V \mathbf{N}_i(\mathbf{r}) \cdot \mathbf{R}\left(\epsilon_0 \hat{\epsilon}_\infty\right) \mathbf{R}^{\top} \mathbf{N}_j(\mathbf{r}) \,\mathrm{d}V \\
T^{\sigma}_{i,j} &= \iiint\limits_V \mathbf{N}_i(\mathbf{r}) \cdot \mathbf{R} \hat{\sigma} \mathbf{R}^{\top} \mathbf{N}_j(\mathbf{r}) \,\mathrm{d}V \\
P_{i,j} &= \iint\limits_{S_{ABC}} \Big[\vec{n} \times \mathbf{N}_i(\mathbf{r})\Big] \cdot \Big[\mathbf{R}^{\top^{-1}} \frac{1}{\mu_0 \hat{\mu}_r} \mathbf{R}^{-1} \nabla \times \mathbf{N}_j(\mathbf{r})\Big] \,\mathrm{d}S \\
Q_{i,j} &= \iint\limits_{S_{ABC}} Y_0 \Big[\vec{n} \times \mathbf{N}_i(\mathbf{r})\Big] \cdot \Big[\vec{n} \times \mathbf{N}_j(\mathbf{r})\Big] \,\mathrm{d}S \qquad (2.1\mathrm{c}) \\
\Phi^{0}_{i,j} &= \sum_{p=1}^{P} G_p \epsilon_0 \left(1 - e^{-\frac{\Delta t}{2\tau_p}}\right) \iiint\limits_V \mathbf{N}_i(\mathbf{r}) \cdot \mathbf{R}\left(\hat{\epsilon}_{s,p} - \hat{\epsilon}_\infty\right) \mathbf{R}^{\top} \mathbf{N}_j(\mathbf{r}) \,\mathrm{d}V \\
\Phi^{1/2}_{i,j} &= \sum_{p=1}^{P} G_p \epsilon_0 \left(1 - e^{-\frac{\Delta t}{2\tau_p}}\right) \left(e^{-\frac{\Delta t}{2\tau_p}}\right) \iiint\limits_V \mathbf{N}_i(\mathbf{r}) \cdot \mathbf{R}\left(\hat{\epsilon}_{s,p} - \hat{\epsilon}_\infty\right) \mathbf{R}^{\top} \mathbf{N}_j(\mathbf{r}) \,\mathrm{d}V \\
\tilde{f}_i &= \iiint\limits_V \mathbf{N}_i(\mathbf{r}) \cdot \frac{\partial \mathbf{J}_{imp}(\mathbf{r}, t)}{\partial t} \,\mathrm{d}V \\
\left(\tilde{f}^{w}_{n}\right)_i &= \int\limits_s \left(N^{w}_{n}\right)_i(s) \frac{\partial V_{n,imp}(s, t)}{\partial t} \,\mathrm{d}s
\end{aligned}
$$

Note that the first two rows of the final system of equations 2.1a handle the basic electric field behavior and its coupling with wire currents. The following row (containing $\mathbf{v}_{inc}$) takes care of the excitation defined by a boundary condition. Vector $\mathbf{f}$ represents current source and $\mathbf{g}$ stands for voltage source. Vectors $\mathbf{f}$ and $\mathbf{g}$ are set to zero by default (standard BUTFE code does not consider such sources). Matrix $\mathbf{\Phi}$ and vector $\mathbf{\Psi}$ are related to dispersive media. Matrix $\mathbf{R}$ defines rotation of a material with respect to the global coordinate system.

The boundary excitation $\mathbf{E}_{inc}(\mathbf{r}, t)$ is approximated as

$$
\tilde{\mathbf{E}}_{inc}(\mathbf{r}, t) \approx \sum_{j=1}^{6} \mathbf{N}_j(\mathbf{r}) \left(e_{inc}(t)\right)_j \qquad (2.2)
$$

which may cause some inaccuracy. However passing analytically computed results of terms $\nabla \times \mathbf{E}_{inc}(\mathbf{r}, t)$ and $\vec{n} \times \mathbf{E}_{inc}(\mathbf{r}, t)$ as inputs of the solver would complicate implementation into the framework significantly.

Approximation of the magnetic field over an element is given by

$$
\tilde{\mathbf{H}}(\mathbf{r}, t) \approx -\mathbf{R}^{\top^{-1}} \frac{1}{\mu_0 \hat{\mu}_r} \mathbf{R}^{-1} \sum_{j=1}^{6} \nabla \times \tilde{\mathbf{N}}_j(\mathbf{r}) \int\limits_t e_j(t) \,\mathrm{d}t \qquad (2.3)
$$

2.4 FEM approximation

Approximation of desired quantity is a key point of the finite element method. The integral of an unknown field quantity is replaced by a multiplication of unknown coefficients and integrals of the basis functions which are easy to determine.

The BUTFE solver employs 1D nodal elements for an approximation of wire currents and 3D edge elements for an approximation of the electric field.

Aproximation in 1D

One-dimensional approximation is used for discretization of wires. Wires are discretized into a set of line segments. Each segment (element) is expressed in terms of simplex coordinates

$$\xi_1(s) = \frac{l_{P2}}{l_{12}} \quad \xi_2(s) = \frac{l_{1P}}{l_{12}} \quad s \in [1,2] \tag{2.4}$$

where P is the point of interest on the line segment and l denotes the length of a line formed by given nodes. It is clear that the i-th simplex coordinate takes value 1 at the i-th node, vanishes on the other one, $\sum_{i=1}^{2} \xi_i(s) = 1$ and $\xi_i(s) \in [0,1]$.

Shape function is a node-oriented scalar function defined as follows:

$$\tilde{N}_1^w(s) = \xi_1(s), \quad \tilde{N}_2^w(s) = \xi_2(s) \tag{2.5}$$

Approximation in 3D

The structure is discretized into a set of tetrahedrons where each tetrahedron is expressed in terms of simplex coordinates. The local simplex coordinate system $\zeta_{1..4}$ satisfies for a given tetrahedron

$$\begin{aligned} \zeta_1(\mathbf{r}) &= \frac{V_{P234}}{V_{1234}} \quad \zeta_2(\mathbf{r}) = \frac{V_{1P34}}{V_{1234}} \\ \zeta_3(\mathbf{r}) &= \frac{V_{12P4}}{V_{1234}} \quad \zeta_4(\mathbf{r}) = \frac{V_{123P}}{V_{1234}} \quad \mathbf{r} \in V \end{aligned} \tag{2.6}$$

where P is the point of interest inside the tetrahedron and V denotes the volume of a tetrahedron formed by given nodes. It is clear that the i-th simplex coordinate takes value 1 at i-th node, vanishes on the opposite face, $\sum_{i=1}^{4} \zeta_i(\mathbf{r}) = 1$ and $\zeta_i(\mathbf{r}) \in [0,1]$.

The simplex coefficients $a_1..d_4$ preserve the following relation between the simplex and Cartesian system:

$$\zeta_i(\mathbf{r}) = a_i + b_i P^x + c_i P^y + d_i P^z \quad \mathbf{r} \in V \tag{2.7}$$

where P^x, P^y and P^z are the Cartesian components of the point of interest.

Shape function is edge-oriented vector function defined as follows:

$$\tilde{\mathbf{N}}_k(\mathbf{r}) = l_{i,j}\Big[\zeta_i(\mathbf{r})\nabla\zeta_j(\mathbf{r}) - \zeta_j(\mathbf{r})\nabla\zeta_i(\mathbf{r})\Big] \quad k = 1..6 \tag{2.8}$$

where $l_{i,j}$ is the length of the k-th edge going from the i-th towards the j-th node. The shape function takes on the remaining (dotted) edges only the perpendicular component (or zero value). A tangential component along any edge can be set only using the coefficient and shape function belonging to that edge. Since the relevant coefficient value is shared between adjacent elements, tangential continuity of the computed field is guaranteed.

See [6] for further information about obtaining simplex coefficients and computation with basis functions.

2.5 Conclusion

Chapter 2 has been dedicated to the TDFE solver BUTFE and its excitation module BUTFE_EXC, including the description of the semantic location within the computational framework HIRF-SE.

Based on a *three-dimensional* (3D) nodal-based FEM code for solving modal analysis taken from my master thesis, BUTFE has evolved into a 3D edge-based TDFE solver with significant improvements in excitation, applicable material properties and boundary conditions.

Further development is facilitated thanks to a well-arranged definition of matrices of the final system 2.1. Each matrix is stored in a stack. Accompanying details contain information like which coupling matrices to multiply by (general/wire), which matrix generator to use ($\mathbf{S}/\mathbf{T}$ in 1D, $\mathbf{S}/\mathbf{T}/\mathbf{P}/\mathbf{Q}$ in 3D) or which temporal operation to apply (none/derivative/second derivative/integral/double integral). The program passes through the stack, automatically applies a given temporal scheme and puts the resulting matrices in the proper place in the system of equations.

BUTFE_EXC is a GUI application that allows to set excitation coefficients related to a given layer (mesh group) of the computational mesh. Excitation can be set in three ways: by formula, by formula considering a perforated slab (using PEMF) or waveguide excitation.

Considering the nature of the chapter content, most of the work presented here is rather of applied nature. The solver (module) is important from three main perspectives:

- **HIRF-SE framework:** The module offers results obtained by another method. Results obtained by various methods can give a better overview of the solved problem.
- **fundamental research:** The solver offers basic functionality which forms the basis for further research activities.
- **applied research:** The solver represents a good starting point for further applied industrial research.

Although the BUTFE solver can still be considered as underdeveloped (only basic order of approximation, first order ABC only, no *perfectly matched layer* (PML)), significant steps forward have been made. Future students of this faculty will be able to focus on more attractive topics of numerical analysis having a functional program core already available.

3 BENT WIRES

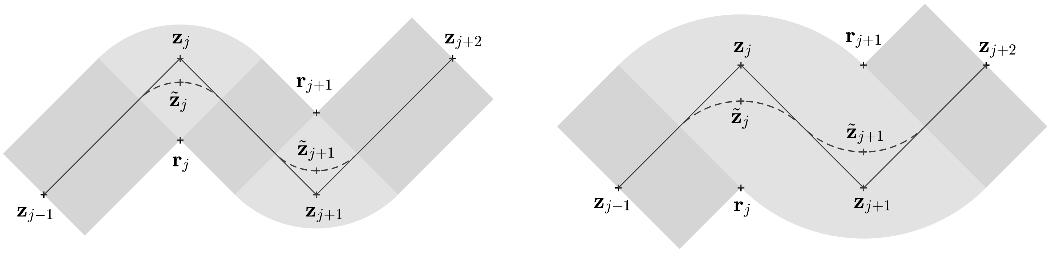
Chapter 3.1 describes shortcomings of already published method of a thin-wire approximation of bent wires and briefly introduces suggested improvements. Chapter 3.2 defines methods for an automatic mesh correction of the wire mesh. Chapter 3.3 concludes this topic.

3.1 Proposed approach

The solution for arbitrarily oriented wires is given in [5]. Authors divide the wire into a set of straight parts and bends. While the straight parts follow the original line between subsequent nodes, bends require special treatment. In order to keep continuity between two subsequent straight parts, the centerline of the interpolation cylinder does not pass exactly through a given node.

Figure 3.1a shows this approach applied on two 90° bends. The wire segment is described by a mesh of four nodes $\mathbf{z}$. One may see that the centerline of the interpolation cylinder matches the wire mesh in the case of the straight parts but goes rather through a virtual point $\tilde{\mathbf{z}}$ in the case of a bent part.

Note that the bends are naturally three-dimensional, however the test bends are performed in a single plane for easier presentation of the results.



(a) $r_0 = 0.3\,\|\mathbf{z}_j\mathbf{z}_{j+1}\|$ **(b)** $r_0 = 0.5\,\|\mathbf{z}_j\mathbf{z}_{j+1}\|$

Fig. 3.1 *The interpolation cylinder proposed in [5]*

As authors admit, they assume that two neighboring sectors are not overlapping each other. Such an overlap can be caused by a large radius of the interpolation cylinder r_0 in the case of sharp bends and an inappropriate choice of discretization of the wire with respect to the surrounding area. Note that the radius r_0 is given by discretization of the surrounding area. Figure 3.1b shows the limit case when bent parts are touching each other. An increase of r_0 or a sharper bend causes overlaps of the wire segments.

It is usually difficult to discover the discretization issue because the approach solves each bend separately without any knowledge about position of the following bend and thus the possibility to detect possible overlaps. The modeler would probably have to start an iterative process of meshing and check of the continuity of the interpolation cylinder visually.

I have focused on inventing another approach which would handle the discretization issue by itself. While the previous approach focused on the straight parts and treated the bends to create a smooth cylindrical object, I have decided to do the opposite. I have arranged that the centerline of the bend parts is passing through nodal points. The straight parts are then treated to match the bends and create a smooth cylinder (see figure 3.2).

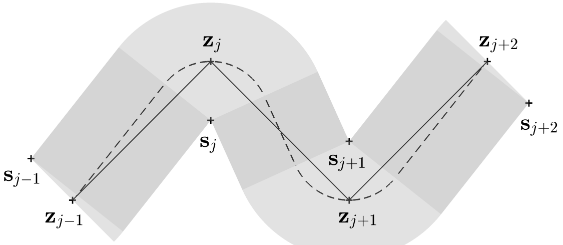


Fig. 3.2 *The interpolation cylinder, proposed apporach,* $r_0 = 0.3\,\|\mathbf{z}_j\mathbf{z}_{j+1}\|$

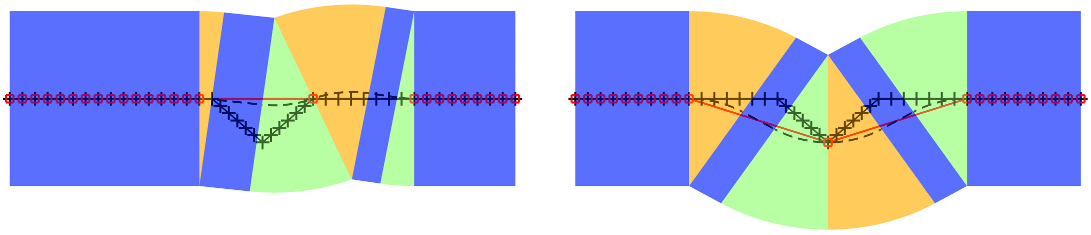
As it turned out, this approach is sensitive to a large r_0 as well. However, it is able to detect overlaps of the neighboring segments. The detection allows to skip one or more nodes causing the overlap(s).

The advantage of this approach is that it solves a pair of two subsequent bends together. Unsolvable geometry produces imaginary angle values which are easy to detect. The user can be asked to increase the density of the tetrahedral mesh in proximity of the wire. A second option is to slightly modify the wire mesh in order to avoid intersections. This avoids user interaction, however it may lead to loss of accuracy. The following chapter shows a proposal of how to deal with the modification of the wire mesh.

3.2 Construction strategies

The proposed approach of the construction of the cylindrical area between two consecutive nodes gives information (a feedback) whether a wire segment is feasible or not. We are going to present two strategies which use the feedback to make a subset of the original nodes. This subset contains non-conflicting nodes which form seamless cylindrical area.

The easiest strategy could be called *avalanche processing.* It means that we sequentially process segments from one end of the wire to the other.



(a) *Avalanche processing* **(b)** *Progressive refinement*

Fig. 3.3 *Results of two construction strategies*

Another option is to use so-called *progressive refinement.* This strategy is more complicated but offers better approximation of the original shape. Nodes are processed in according to their importance where the most important node defines the sharpest bend of the wire. See figure 3.3 for comparison of these two strategies. The original mesh is denoted by black color, nodes of the final segments are marked by red squares. Centerline of the resulting cylindrical area is denoted by black dashed line. Part BI is colored in orange, BII in green and S is colored in blue.

3.3 Conclusion

The proposed approach described in chapter 3 aims to improve thin wire approximation of bent wires published in [5]. The improvement involves different procedure of creation of cylindrical area surrounding the wire. Core functionality of the thin-wire model remains the same.

Main advantage of the proposed approach lies in presence of the feedback. The approach allows to detect and prevent overlaps within a wire segment (between two consecutive nodes). The feedback can recommend the user to modify computation mesh or it can be used for an automatic mesh correction during build process of the cylindrical area.

The construction strategies have been developed for automatic handling of situations where the original approach fails. Two strategies for the automatic mesh correction have been suggested: *avalanche processing* and *progressive refinement*. Since the second method exhibited better performance, *avalanche processing* was no longer applied.

While focusing on the continuity of the cylindrical area, the resulting centerline may not follow the original route exactly. This brings some loss of accuracy. However user may be warned about the inconsistency (by number of omitted nodes) and take reasonable measures.

Unfortunately, the loss of accuracy given by overlaps and errors (Edelvik et al.) can be hardly compared with the loss of accuracy given by a different route of the cylinder and omitted nodes (proposed approach). Current distribution cannot be compared due to unavailability of measurements or any validated software, especially due to inability to ensure same conditions (boundary conditions and excitation).

4 FILE INTERFACE

This chapter is devoted to description of the file interface between modules in the HIRF-SE framework. Chapters 4.1 and 4.2 describe connection of the modules BUTFE_EXC and BUTFE. Chapter 4.3 introduces Amelet-HDF C library acting as an input wrapper (used in the BUTFE module). Finally, chapter 4.4 concludes current state of the Amelet-HDF specification and the library.

4.1 Module BUTFE_EXC

The Amelet-HDF format is based on *Hierarchical Data Format* (HDF) [3]. Each object is described by a path starting with "/". This symbol represents the file root, another "/" symbols indicate one step down in the tree structure (e.g. /mesh/mesh_butfe).

Entry point of the simulation is *input.h5:/simulation/sim_exc*. The dataset *inputs* contains path to all inputs: only the input mesh */mesh/mesh_gid* in this case. The dataset *outputs* contains paths to all requested outputs: */mesh/mesh_butfe*, */globalEnvironment/-global environment* and */electromagneticSource/sourceOnMesh/electromagnetic source.*

The output file looks similarly. It contains only outputs, so that *mesh_gid* is no longer present. Note that there is a new layer *EXC_eg* in *mesh_butfe* created by the excitation module from layer *EXC*. The excitation coefficients */electromagneticSource* and the list of time samples */globalEnvironment* are filled by correct values of excitation coefficients and time samples based on the user definition of the excitation.

4.2 Module BUTFE

Entry point of the simulation is *input.h5:/simulation/sim.*

Path */simulation/sim/parameter* is determined to contain parameters of the simulation. In this case, it contains three attributes: *maxThreads*, *preconditioner* and *solver.* The first one sets the maximal number of threads to be used for the computation. The default value (zero) sets the number of threads equal to the number of cores available in the system. Remaining attributes pick a combination from a list of preconditioners/solvers of the mathematical library Lis used for the computation [15].

User has defined four *output request* instances. Each instance defines outputs of the simulation and contains *subject* (physical quantity to compute), *object* (where to compute that quantity) and *output* (storage for the result). The resulting objects are situated in */floatingType* category.

Subjects of the *output request* instances are stored in the */label/output request* dataset. In the simulation example, the dataset contains strings "electricField" and "magneticField".

The category */physicalModel* defines materials of the model. The *ABC* material is reserved for absorbing boundary condition. In normal situation, material parameters would be set via attributes of groups *electricConductivity*, *magneticConductivity*, *relativePermeability*, and *relativePermittivity*.

Instances of the */link* category couple any two objects together. Each instance contains *subject* (material/object) and *object* (place). Note that material ABC has to be coupled with excitation layer *EXC* as well in order to achieve proper form of the equation system.

The output file differs only in content of the */floatingType* category. The */floatingType* now contains results of the simulation.

The output file must contain only updated */floatingType*, */mesh* and */simulation* categories to allow the framework to read results back to its database. BUTFE is forming the output file by copying the input one and replacing the empty data structures in */floatingType* by the results. This allows to save both results and all input data for the simulation in a single file.

Focusing on the objects in */floatingType* category, each of them contains three-dimensional dataset *data* containing the field values. Group *ds* includes three datasets containing description of particular dimension of the *data*. Dataset *dim1* stores path to given mesh group (e.g. */mesh/mesh_butfe/unstructuredmesh/group/DOMAIN1*), *dim2* describes components of the field (“x”, “y”, “z”) and *dim3* contains time samples of the *data*.

4.3 Amelet-HDF C library

The AxesSim company, a developer of the Amelet-HDF format, created a set of source codes that should help module developers to employ this new file format. However, it appeared that it contained several bugs and memory leaks. So the code was developed rather from scratch instead of using developer’s one. Unique orientation of the new code has offered possibility to implement the code as a library and make it more universal so that other partners could benefit from it (although it did not cover all the possibilities of the Amelet-HDF format). The resulting code of the library was uploaded into the source repository of the Amelet-HDF website into a separate branch */svn/amelethdf-c/branches/vlse* [2].

The library provides all necessary functions for reading input files mentioned in chapter 4.2 and some extra reading functions which may be valuable for other participants of the project or people who decide to use Amelet-HDF. Since writing capabilities of the solver were pretty well covered by the HDF functions, there was no need to implement them into the library.

The code is divided into files so that each file covers particular category of the Amelet-HDF (except the general purpose ones). Each file contains functions for reading and printing data from given Amelet-HDF file. Functions for unallocating memory occupied by the read data are included too. Names of the library functions start with library prefix “AH5_” followed by “read_”, “print_” or “free_” in order to suggest above-mentioned purpose.

The Amelet-HDF contains several levels of the tree structure. For example, */outputRequest* category contains various number of groups. Each group contains various number of instances. The library contains functions for accessing each level individually. For example, the function *AH5_read_outputrequest()* reads whole */outputRequest* category. It reads number of groups and calls *AH5_read_ort_group()* to get contents of the instance. Each *AH5_read_ort_group()* reads number of instances and calls *AH5_read_ort_instance()* to get their contents. A programmer can use any of these functions to work on specific level.

The library includes following features of the Amelet-HDF specification:

- sources: *planeWave*, *sphericalWave*, *generator*, *dipole*, *antenna* and *sourceOnMesh*
- all kinds of floatingTypes: *singleInteger*, *singleReal*, *singleComplex*, *singleString*, *vector*, *rationalFunction*, *generalRationalFunction*, *rational*, *dataSet*, *arraySet*
- *mesh* (both types: *structured* and *unstructured*) including *selectorOnMesh* and *meshLink*
- *physicalModel*: *volume material*, *anisotropic*, *surface materials*, *interface*
- *exchange surface*, *external element*, *globalEnvironment*, *label*, *link*, *localizationSystem*, *outputRequest*

4.4 Conclusion

This chapter has been devoted to more detailed explanation of encapsulation of the BUTFE modules inside the HIRF-SE platform. The Amelet-HDF has proven to be useful format for storing data of the electromagnetic simulations (although there is still some room for improvements). Many developers of EM software are using their own formats which brings compatibility issues.

It should be said that the Amelet and especially all the HDF stuff behind may be difficult to understand and limiting when comparing to developing own data format. An overhead connected with the implementation of a new format may be significant, especially for smaller projects. This is exactly the case where the library presents main benefit. It makes the process of the new format understanding and implementation much quicker.

Original contribution of the work mentioned in this chapter consist in encapsulated design of the Amelet-HDF C library and its data structures.

Besides its original use in the BUTFE module, the library was used in modules ANN-Training and ANN-Estimation which were developed for solving postprocessing issues using artificial neural networks in the HIRF-SE framework. The library has recently taken place of the main C library in the Amelet-HDF project. The AxesSim company (developer of the Amelet-HDF specification) has decided to incorporate the library in their software and still enhances its functionality.

5 DEVELOPMENT TOOLS

This chapter deals with description of tools that had to be created in order to simplify development and testing of the modules. Chapter 5.1 describes a comparison tool CMP that compares results with a reference. Chapter 5.2 is devoted to a mesh inspector/post-processor called Visualizer and finally 5.3 concludes this topic.

5.1 CMP

The BUTFE module employs OpenMP *application program interface* (API) that supports a shared-memory parallelism [16]. Development of such applications can be tricky. Apart from programming mistakes known from a single-threaded programs, the OpenMP programs can contain additional issues caused by broken thread safety. Some of the thread safety issues can be observed only once per tens of simulations. Performing such a number of simulations and checks manually would require large amount of time. However by using a comparison tool, one may run the simulations in a loop and check resulting comparison of all iterations at once.

The comparison tool examines all simulation outputs of an output Amelet-HDF file with outputs of a reference output file (obtained from a previous simulation declared to be valid). Since the BUTFE module employs an iterative solver, results may vary a little. Therefore a number of samples with relative deviation greater than some limit is observed.

5.2 Visualizer

Almost all of the input data for *finite element method* (FEM) are closely connected with the computational mesh. The mesh is formed by a complex set of data which is not human readable without any visualization. The visualization is usually provided by a meshing software, which allows to check validity of the mesh right after its creation. However the mesh data format is converted several times before the computation (mainly due to the Amelet-HDF format).

The Visualizer shows the mesh data right before the computation. It shares its mesh import code with the BUTFE so it operates with same data as the solver. This helps avoiding potential errors caused by wrong mesh import and visualizing other variables related to the solver (global edges, outputs).

In fact, the Visualizer tool does the job of the module Paraview or post-processing mode of the module GiD mentioned in chapter 2.1. The reason for developing the in-house visualization tool was due to the fact that the GiD did not work at all with the BUTFE outputs and the Paraview module had usually crashed. The Visualizer is even better because it can work in the course of running simulation while the others would have to wait for the end of the simulation to post-process the output data.

The tool operates in three modes. The first one is a mesh inspector only. It is called by the Visualizer executable with a name of an Amelet-HDF file as an input parameter.

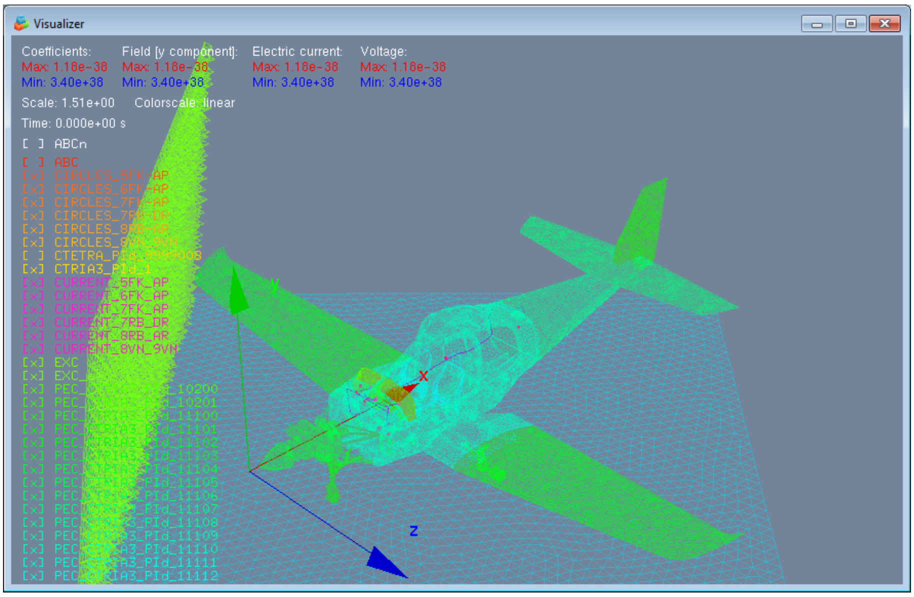

Coefficients: Max: 1.18e-38 Min: 3.40e+38
Field [y component]: Max: 1.18e-38 Min: 3.40e+38
Electric current: Max: 1.18e-38 Min: 3.40e+38
Voltage: Max: 1.18e-38 Min: 3.40e+38
Scale: 1.51e+00 Colorscale: linear
Time: 0.000e+00 s

Fig. 5.1 *Visualizer: mesh inspector.*

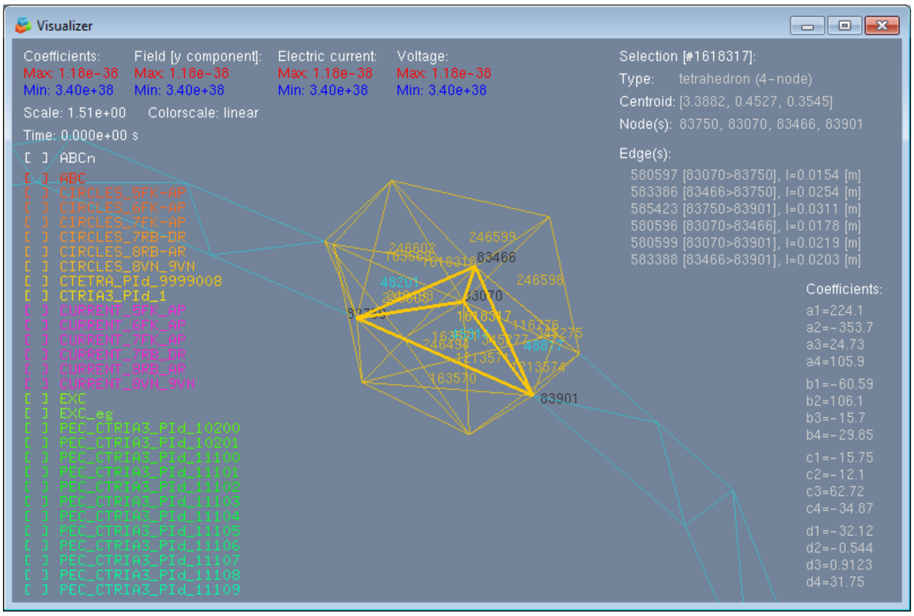
The second mode is a combined inspector/post-processor. The Visualizer executable is called with no parameters. After the execution, the visualization tool waits for the execution of the solver (BUTFE module). Once the solver is running, the visualization tool starts displaying the mesh and output requests of the ongoing simulation (only one simulation is expected to running at a time). If the simulation is already running, visualization starts immediately.

The data transfer between both the programs is carried out using *named pipes* [17]. Although this *inter-process communication* (IPC) is not "naturally cross-platform", it is relatively simple and applicable on both the Linux and Windows compilers used for the BUTFE module. The pipe data contain name of the input file of the solver to be read by the visualization tool. The mesh data are stored in the system memory twice. Once for the solver and once for the visualization tool. It may look like a wasting of resources but on the other hand, both the tools must be able to work independently. Rest of the pipe data contains the post-processing data of the output requests for last computed time step. The BUTFE module can work independently considering a small overhead caused by the IPC (it sends the pipe data regardless of a receiving).

The third mode is post-processor only. It allows to open already saved output file containing results. Unlike the second mode, there is no connection with the solver and no simulation is running. The visualization tool allows to move in time back and forth within given data range of the file.

The tool allows to toggle visibility of elements and/or element numbers of particular mesh group, switch between filled and wireframe mode and set type of coloring of the field arrows. The above mentioned options are accessible through a menu by pressing right mouse button. Some of them are also accessible using a keyboard.

The user can highlight particular element and its neighborhood. The neighborhood



```
Coefficients:      Field [y component]:  Electric current:  Voltage:
Max: 1.18e-38      Max: 1.18e-38         Max: 1.18e-38      Max: 1.18e-38
Min: 3.40e+38      Min: 3.40e+38         Min: 3.40e+38      Min: 3.40e+38

Scale: 1.51e+00    Colorscale: linear

Time: 0.000e+00 s
```

```
Selection [#1618317]:
Type:     tetrahedron (4-node)
Centroid: [3.3882, 0.4527, 0.3545]
Node(s):  83750, 83070, 83466, 83901

Edge(s):
  580597 [83070>83750], l=0.0154 [m]
  583386 [83466>83750], l=0.0254 [m]
  585423 [83750>83901], l=0.0311 [m]
  580596 [83070>83466], l=0.0178 [m]
  580599 [83070>83901], l=0.0219 [m]
  583388 [83466>83901], l=0.0203 [m]

Coefficients:
a1=224.1
a2=-353.7
a3=24.73
a4=105.9

b1=-60.59
b2=106.1
b3=-15.7
b4=-29.85

c1=-15.75
c2=-12.1
c3=62.72
c4=-34.87

d1=-32.12
d2=-0.544
d3=0.9123
d4=31.75
```

Fig. 5.2 *Visualizer: element inspection.*

is formed by all edges, triangles, and tetrahedrons touching selected element. Important information about the element is displayed in the top right area of the window.

Particular global edge of the mesh can be located. This is important for the situation, when placement of the given edge coefficient is examined.

The visualization tool uses OpenGL [18] toolkit FreeGLUT [19]. Display lists, vertex arrays and vertex buffer objects are used in order to improve performance of the code [18].

5.3 Conclusion

The development process showed the need for some new tools to be designed simultaneously with the solver creation. They were not requested by the project, but they made the development more efficient.

The tool CMP became valuable in testing of parallel code. It can be used in the future for automatic comparison of two different approaches implemented in single module or comparison of two different modules giving same kind of results.

The Visualizer had proven its usefulness in many occasions. At the beginning, it was valuable tool for checking intersections between given volumetric mesh and objects inside (objects must be part of the volumetric mesh). GiD did not provide such an option.

The visualization of the results of the ongoing simulation allows to observe evolution of the simulation process. The user can stop the simulation in the case of any problem or when all important phenomena have ended up earlier than expected.

The visualization of the results of the finished simulation allows to skip uninteresting phase of the simulation and slowly scroll through the interesting one.

6 CONCLUSION

This work deals with four main topics where three of them are of rather applied nature and one provides an improvement of a thin wire approximation in FEM.

The applied nature of the work consist in development of various programs and libraries aiming to implement a TDFE solver into the HIRF-SE framework. This required designing unique development tools and proposed several innovative solutions. The development of the module was being complicated due to the following problems:

- the HIRF-SE framework became available too late
- the need to rewrite the code into the C language while it was originally written in the MATLAB$^{©}$
- lack of any excitation tool
- lack of any suitable post-processing tool
- problem with implementation of the time-domain PML into the solver

Since many of the HIRF-SE code developers joined the project with already finished programs, their main task was only to adapt their solver to the framework by adding support of the Amelet-HDF format. On the other hand, the BUTFE module has originated almost from scratch during the project lifetime. Therefore, the Amelet-HDF format became native input/output format of the module. BUTFE required input data which should be generated by the HIRF-SE framework (which was not available at that time). This issue was solved by programming a substitute code for assembling input data for the solver.

The need to rewrite the code into C language was probably caused by some misunderstanding in the requirements during initial phase of the project.

Definition of an excitation for the solver was expected to be handled by the framework itself. The solver had to be a commandline tool only and therefore it would be impossible (extremely user-unfriendly) to define excitations without any GUI. A new module BUTFE_EXC was invented to deal with this problem.

The lack of suitable post-processing tool was solved by programming of the Visualization tool mentioned in chapter 5.2.

The PML was intended to form a reflectionless boundary in the BUTFE. However, all attempts to implement the PML failed.

Despite all the mentioned complications, the module was succesfully implemented into the framework as an optional module. The Amelet-HDF library (chapter 4.3) recorded success while it became an official C library of the Amelet-HDF project.

The improvement of the thin wire approximation (chapter 3) lies in different procedure of construction of the cylindrical area around the wire, core of the method remains the same. The proposed procedure can detect and prevent overlaps of neighboring wire segments. The feedback allows implementing an automatic omission of certain nodes and creation of a geometry that approximates the original wire mesh most closely. This leads to more accurate results provided by the numerical solver. The method cannot prevent overlaps caused by proximity of two or more nodes that are not strictly next to each other in the wire mesh. However, occurrence of such situation is less probable than occurrence of overlaps caused by two consecutive nodes. A general algorithm that would avoid intersection caused by non-consecutive nodes could be subject of a future investigation.

Experiences were presented on the International Travelling Summer School on Microwaves and Lightwaves [7], International Conference on Electromagnetics in Advanced Applications [8] and on the COST workshop [9]. Integration and validation of the TDFE module was presented in a joint paper prepared with colleagues from University of Gdansk and Sapienza University of Rome [10].

BIBLIOGRAPHY

[1] *HIRF SE High Intensity Radiated Field Synthetic Environment* [online]. 2008 [cit. 2011-04-14]. Available at WWW: <http://www.hirf-se.eu>.

[2] *amelet-hdf - Tool development for Amelet-HDF Specification. - Google Project Hosting* [online]. 2009 [cit. 2011-04-26]. Available at WWW: <http://code.google.com/p/amelet-hdf/>.

[3] *The HDF Group : Information, Support, and Software* [online]. 1988 [cit. 2011-04-26]. Available at WWW: <http://www.hdfgroup.org/>.

[4] ŠTUMPF, M.; DE HOOP, A.; LAGER, I. Closed-Form Time- Domain Expressions for the 2D Pulsed EM Field Radiated by an Array of Slot Antennas of Finite Width. In *Symposium Digest - 20th International URSI Symposium on Electromagnetic Theory.* Berlin: International Union of Radio Science, 2010. p. 786-789. ISBN: 978-1-4244-5154- 8.

[5] EDELVIK, F.; LEDFELT, G.; LÖTSTEDT, P.; RILEY, D. J. An Unconditionally Stable Subcell Model for Arbitrary Oriented Thin Wires in the FETD Method. *IEEE Transactions on Antennas and Propagation.* 2003, vol. 51, no. 8, p. 1797-1805.

[6] SAVAGE, J.S.; PETERSON, A.F. Higher-order vector finite elements for tetrahedral cells. *IEEE Transactions on Microwave Theory and Techniques.* June 1996, vol. 44, no. 6, p. 874-879.

[7] ŠEDĚNKA, V. Advanced finite element method. In *Proceedings of 20th International Travelling Summer School on Microwaves and Lightwaves.* 2010, p. 1-8.

[8] CIGÁNEK, J.; KADLEC, P.; RAIDA, Z.; ŠEDĚNKA, V. Finite Element Analysis of Composite Objects: Time Domain Versus Frequency Domain. In

[9] KADLEC, P.; RAIDA, Z.; ŠEDĚNKA, V.; CIGÁNEK, J. Time domain finite element analysis of electrically large composite objects. In

[10] ŠEDĚNKA, V., CIGÁNEK, J., KADLEC, P., RAIDA, Z., WIKTOR, M., SARTO, M.S., GRECO, S. Time-Domain Finite Elements for Virtual Testing of Electromagnetic Compatibility. *Radioengineering.* 2013, vol. 22, no. 1, p. 309-317.

[11] *MathWorks - MATLAB and Simulink for Technical Computing : MATLAB Compiler - MATLAB* [online]. 2011 [cit. 2011-12-28]. MATLAB Compiler. Available at WWW: <http://www.mathworks.com/products/compiler/>.

[12] *MathWorks - MATLAB and Simulink for Technical Computing : Working with the MCR :: Deployment Process (MATLAB© Compiler^{TM})* [online]. 2011 [cit. 2011-12-28]. Working with the MCR. Available at WWW: <http://www.mathworks.com/help/toolbox/compiler/f12-999353.html>.

[13] *www.gidhome.com* [online]. 2011 [cit. 2011-12-30]. What's GiD. Available at WWW: <http://gid.cimne.upc.es/>.

[14] *ParaView - Open Source Scientific Visualization* [online]. 2011 [cit. 2011-12-30]. ParaView. Available at WWW: <http://www.paraview.org/>.

[15] *SSI: Scalable Software Infrastructure for Scientific Computing: Lis: a Library of Iterative Solvers for Linear Systems* [online]. 2011 [cit. 2011-12-31]. Available at WWW: <http://www.ssisc.org/lis/index.en.html>.

[16] *OpenMP.org*: *The OpenMP® API specification for parallel programming* [online]. 2012 [cit. 2012-05-01]. Available at WWW: <http://www.openmp.org/>.

[17] *Named pipe - Wikipedia, the free encyclopedia*: *Named pipe* [online]. 2012 [cit. 2012-05-01]. Available at WWW: <http://en.wikipedia.org/wiki/Named_pipe>.

[18] *OpenGL - The Industry Standard for High Performance Graphics* [online]. 2012 [cit. 2011-05-02]. Available at WWW: <http://www.opengl.org/>.

[19] *The freeglut Project :: About* [online]. 2012 [cit. 2011-05-02]. Available at WWW: <http://freeglut.sourceforge.net>.

CURRICULUM VITAE

Name: Vladimír Šeděnka
Born: August 1983
Contact: vlada.cr@volny.cz

Education

2003-2009	**Technical University of Brno / Department of Radio Electronics** Undergraduate study of Radio Electronics State exam passed in June 2009 Diploma thesis Synthesis of microwave resonators defended in June 2009
2009	**Technical University of Brno / Department of Radio Electronics** Ph.D. study of Radio Electronics State exam passed in May 2011

Experience

2009-2012	**HIRF-SE** developer of a finite-element time-domain solver

Courses

2010	**International Travelling Summer School on Microwaves and Lightwaves** Supélec, Metz (France)

Languages

Czech, English, German